

3772

1

**Final Report for
Cooperative Agreement
NASA Ames Research Center
NASA NCC 2-5274
Developing Information Power Grid Based Algorithms and Software
7/1/98 to 11/1/98
\$50,000**

Principal Investigator
Jack Dongarra
University of Tennessee, Knoxville
Computer Science Department

Technical Officer for the Cooperative Agreement
Subhash Saini
Numerical Aerospace Simulation Systems Branch, T27A-1

Grant Officer
Venoncia Braxton

mid.
MAY 14 1999

CC: 201 A3

CASX

Copied for file

This was an exploratory study to enhance our understanding of problems involved in developing large scale applications in a heterogeneous distributed environment. It is likely that the large scale applications of the future will be built by coupling specialized computational modules together. For example, efforts now exist to couple ocean and atmospheric prediction codes to simulate a more complete climate system. These two applications differ in many respects. They have different grids, the data is in different unit systems and the algorithms for integrating in time are different. In addition the code for each application is likely to have been developed on different architectures and tend to have poor performance when run on an architecture for which the code was not designed, if it runs at all. Architectural differences may also induce differences in data representation which effect precision and convergence criteria as well as data transfer issues. In order to couple such dissimilar codes some form of translation must be present. This translation should be able to handle interpolation from one grid to another as well as construction of the correct data field in the correct units from available data. Even if a code is to be developed from scratch, a modular approach will likely be followed in that standard scientific packages will be used to do the more mundane tasks such as linear algebra or Fourier transform operations. This approach allows the developers to concentrate on their science rather than becoming experts in linear algebra or signal processing. Problems associated with this development approach include difficulties associated with data extraction and translation from one module to another, module performance on different nodal architectures, and others.

In addition to these data and software issues there exists operational issues such as platform stability and resource management. For example, what should happen if some part of the computing platform becomes unavailable or if more resources become available somewhere in the middle of a long run. Checkpointing is typically used by large scale applications to save a state in case something goes wrong. This is a relatively straight forward issue on a single processor for a "reasonably" sized problem but becomes problematic in a highly heterogeneous distributed environment and for problem sizes involving hundreds of gigabytes of data. Some ability to automatically restart an application that has lost processors or to migrate part of an

application to take advantage of computational resources that have become available should be provided. In addition a mechanism for monitoring and steering the application as it progresses would be useful for applications that run for weeks at a time.

Our investigation had three components. The first area of research involved constructing a coupling framework for the environment. One can think of coupling as a mechanism for doing computing on a distributed system. Modules are coupled together to provide functionality, as in the ocean/atmosphere example given above. Work in the coupling framework also encompassed coupling to several individual computing resources, such as a visualization and steering system or to a linear algebra solver. Work on this framework involved building software infrastructure in the form of libraries to allow users to access the framework via an API in the developer's code. Mechanisms for checkpointing and restarting applications, as well as process migration and performance modeling techniques also received attention.

We also examined issues in software and data management. In order to share software and data developed by different research projects some organized, distributed resource management functionality must be provided. We pushed forward the construction of a repository interface for developers. We see the repository as a mechanism for managing data and software developed and used by the community of computational scientists. The system interfaces with the coupled framework described above and are being built initially as an extension of the NETLIB and NetSolve systems. It will be enhanced to address security issues, software compatibility and computational resource issues.

Finally, in addition to these infrastructure aspects, we initiated research on next generation algorithms for the advanced computing environments of interest. We considered providing numerical libraries that are tuned for various architectures. The repository mechanism mentioned above will make users aware of which alternatives are available for a particular machine and the coupling framework will provide the mechanism for accessing the appropriate method.

In this initial study we worked to extend and integrate current systems to develop the new capabilities needed to address the challenges described above. Our research focused on the following areas:

- Parameterizable libraries — We worked to design and construct libraries that are parameterized to allow their performance to be optimized over a range of current and future memory hierarchies. Identifying this range is being done in conjunction with system architects. We chose a few well-understood and important library functions, such as linear algebra, to do this. Target architectures included Clusters of SMPs, Processor in Memory (PIM) such as IRAM <http://iram.cs.berkeley.edu>, and some class of reconfigurable systems, such as BRASS <http://http.cs.berkeley.edu/projects/brass/>.
- Annotated libraries — In collaboration with compiler developers, we focused on identifying opportunities where critical information from the library developer would be supplied to the compiler can aid the compilation process. At the library interface level, this included the memory-hierarchy tuning parameters mentioned above, a performance model that depends on input parameters (e.g. problem size), and tuning parameters. Below the library level, the focus included semantic information, such as dependency information to make LU with pivoting as easy to block as LU without pivoting, and floating point semantic information, such as information to indicate that it is acceptable to reorder certain floating point operations or to handle exceptions in a certain way.

- Sparse linear algebra libraries — Sparse linear algebra benefits from new direct and iterative methods, packaged in object libraries that make them easy to use. There has been a great deal of recent progress in understanding how to exploit memory hierarchies for efficiency (recent supernodal and multifrontal codes) as well as special problem structures (domain decomposition, geometric and algebraic multigrid). In addition to generic libraries, this work was closely tied to particular DOD/DOE applications (like ASCI) to drive development.
- Metacomputing environment. We explored new methods for packaging, distributing, and accessing algorithm technology. This effort also investigated use of network enabled solvers.

In summary, these new libraries speed the transfer of recent algorithmic technology in linear algebra, hierarchical methods, and other areas to high-performance computer users. They foster a new division of labor between compiler writers, library writers and perhaps architects, while at the same time simplifying the problems of all of them. This work enhanced the capability of controlling large complex heterogeneous and dynamic applications over a distributed network supports the assembly of multidisciplinary applications parts of which are built by independent teams.